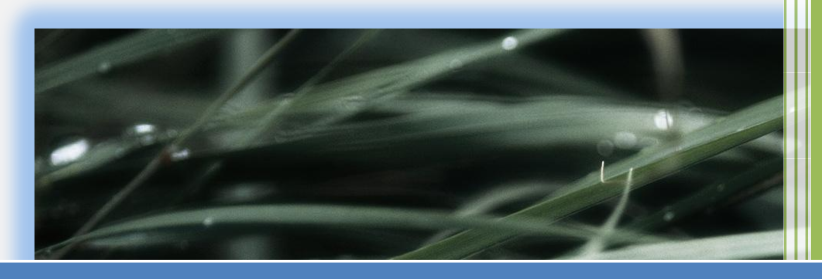
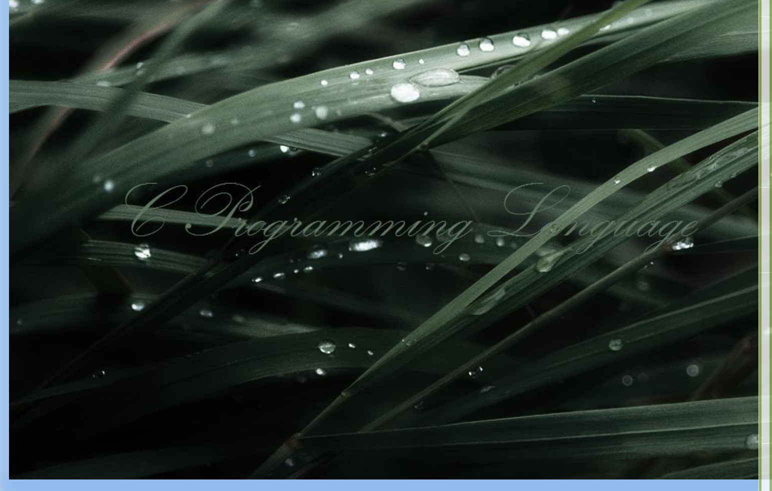
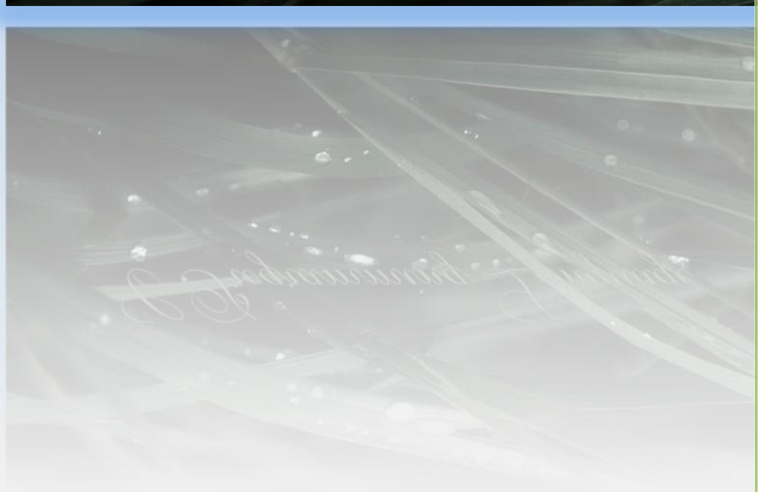


2012

C programming language





OSMAN

Omdurman Ahlia Univiersity

3/29/2012

بسم الله الرحمن الرحيم

C Programming language



لغة سي هي لغة برمجة ابتكرت من قبل دنيس ريتشي و براين كيرنيغان في (AT&T Bell) وهي مختبرات في اوائل السبعينات ، بدأت اللغة باكتساب الشعبية والدعم الواسع الانتشار ، هذا كان لأن لغة سي ليست متوفرة بسهولة للاستعمال التجاري خارج مختبرات بيل ، كما ان لغة السي لها دور كبير في نظام تشغيل اليونيكس هذا النظام الذي طور ايضا في مختبرات بيل كما ان 90% من هذا النظام كتب بلغة السي . كما أنه توجد العديد من المترجمات لهذة اللغة ولكن إذا كنت مبتدىء في هذة اللغة يمكنك تحميل المترجم Turbo C++ IDE من الموقع التالي .

http://www.4shared.com/zip/SiiECQbH/Turbo_C_IDE_Ver_30.htm

مكونات لغة السي :

1/ محرر نصوص لغة سي . Editer

2/ مترجم لغة سي . Compiler

3/ رابط مكتبة سي . Linker library C

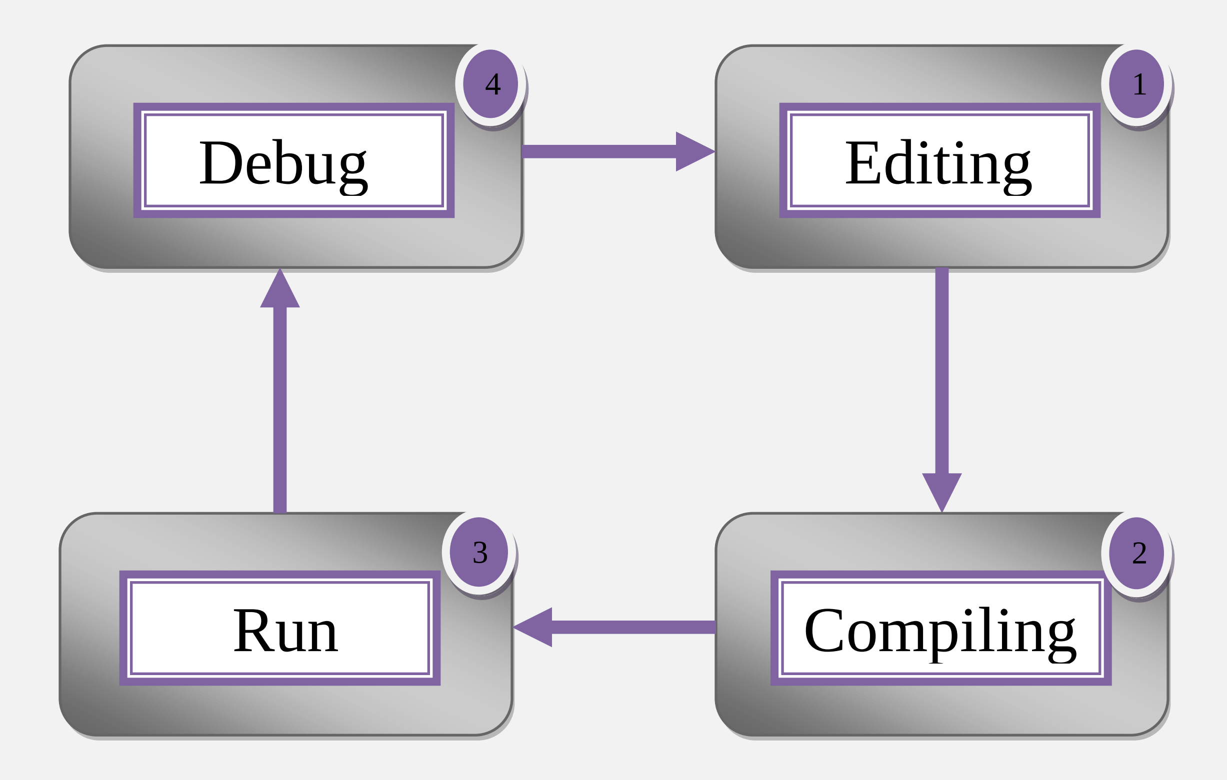
مراحل تنفيذ البرنامج :

1/ مرحلة التحرير : كتابة البرنامج في محرر النصوص . Editing

2/ مرحلة الترجمة : تحويل البرنامج الى لغة اشبه بلغة الألة . Compiling

3/ مرحلة التنفيذ : تحويل البرنامج الى ملف تنفيذي . Run

4/ مرحلة اكتشاف الاخطاء : اذا حدث خطاء في كتابة البرنامج . Debug



الشكل العام للبرنامج في لغة السي :

```
#include<stdio.h>        } توجيهات المترجم
#include<conio.h>
#define pi 3.14  ──────► الإعلان عن ثابت
int add(int x,int y); ──► الإعلان المبدئى لدالة معرفة بواسطة المستخدم
int w; ─────────────────► متغير عام
void min( ) ────────────► الدالة الرئيسية
{
    clrscr( );
    int a,b; ───────────► متغيرات محلية
    printf("\n in put 2num:");   } دوال مكتبية
    scanf("%d%d",&a,&b);
    w=add(a,b); ────────► استدعاء الدالة
    printf("\n w=%d",w);
}
int add(int a,int b) ───► دالة معرفة بواسطة المستخدم
{
    return a+b;
}
```

1/ الدالة main :

هي الاجراء الوحيد اللازم لأي برنامج في لغة السي .

2/ توجيهات المترجم : Compiler directives

هي أوامر تصدر للمترجم ليقوم باداء مهمة محددة .

مثال لتوجيه :

ضم #include

عرف #define

ويمكن تمييز توجيهات المترجم بسهولة لأنها تبدأ بحرف #

3/ ملفات الرأس : Header file

هي ملفات تحتوي على بعض التعريفات المكتبية ، لابد من ضم الملف المناسب عند استخدام أي دالة .

4/ العبارة في لغة سي : Statement

العبارات إما إعلانية تعلن عن شيء ما أو تنفيذية تنفذ مهمة محددة .

معظم العبارات تنتهى بفاصلة منقوطة مع بعض الأستثناءات .

5/ دالة مسح الشاشة ;( )clrscr :

تستخدم لمسح الشاشة ، و clrscr هي اختصار لجملة Clean screen .

6/ الدالة ;( )getch :

تستخدم لتثبيت الشاشة .

المؤثرات Operators

انواع المؤثرات :

1/ مؤثرات حسابية : Arithmetic operators

المؤثر	الأسم	مثال	معناه
++	مؤثر الزيادة	++x	زيادة قيمة المتغير x بواحد اولا إستخدمها ثانيا
		x++	إستخدم قيمة x اولا ثم زدها بواحد ثانيا
--	مؤثر النقصان	--x	انقص قيمة x اولا ثم إستخدمها ثانيا
		x--	إستخدم قيمة x اولا ثم انقصها بواحد
+	الجمع	x+y	اجمع قيمة x الى قيمة y
-	الطرح	x-y	اطرح قيمة y من قيمة x
*	الضرب	x*y	اضرب قيمة x في قيمة y
/	القسمة	x/y	اقسم قيمة x على قيمة y
%	باقي القسمة	x%y	باقي قسمة x على y
=	التخصيص	x=y	ضع قيمة y في x

2/ مؤثرات المقارنة : Relational operators

المؤثر		مثال	معناه
>	اكبر من	x>y	اختبر اذا كانت قيمة x اكبر من y
>=	اكبر من او يساوي	x>=y	اختبر اذا كانت قيمة x اكبر من او تساوي y
<	اقل من	x<y	اختبر اذا كانت قيمة x اصغر من y
<=	اقل من او يساوي	x<=y	اختبر اذا كانت قيمة x اصغر من او تساوي y
==	يساوي	x==y	اختبر اذا كانت قيمة x تساوي y
!=	لا يساوي	x!=y	اختبر اذا كانت قيمة x لا تساوي y

3/ مؤثرات التخصيص المركبة : Complex operator

المؤثر		مثال	معناه
+=	الجمع المركب	x+=y	اجمع قيمة x الى قيمة y . ضع الناتج في x
-=	الطرح المركب	x-=y	اطرح قيمة y من x . ضع الناتج في x
*=	الضرب المركب	x*=y	اضرب قيمة x في قيمة y . ضع الناتج في x
/=	القسمة المركبة	x/=y	اقسم قيمة x على قيمة y . ضع الناتج في x
%=	باقي القسمة المركبة	x%=y	اقسم x على y وضع باقي القسمة في x
-	النفي	-x	انفي قيمة x اذا كانت سالبة وضع موجبة والعكس

4/ مؤثرات منطقية : Logical operators

<table>
<tr><th colspan="2">المؤثر</th><th>مثال</th><th>النتيجة</th></tr>
<tr><td>&&</td><td>AND</td><td>((5>6)&&(2<3))</td><td>F</td></tr>
<tr><td>||</td><td>OR</td><td>((5<6)||(3<2))</td><td>T</td></tr>
<tr><td>!</td><td>NOT</td><td>!(5==6)</td><td>T</td></tr>
</table>

أسبقية المؤثرات الحسابية :

المستوى	المؤثر
الأول	الزيادة او النقصان بعد الاستخدام / النفي .
الثاني	الزيادة او النقصان قبل الاستخدام .
الثالث	الضرب / القسمة / باقي القسمة .
الرابع	الجمع / الطرح .
الخامس	التخصيص / المؤثرات المركبة .

ملاحظات :

1/ اذا احتوى التعبير على اكثر من مؤثر من محتوى مختلف سيتم تنفيذ العملية من مستوى اعلى .

2/ اذا احتوى التعبير على اكثر من مؤثر من نفس المستوى سيتم تنفيذ العملية من اليسار الى اليمين .

3/ الأقواس لها اسبقية أعلى من كل المؤثرات ، بالتالى سيتم تنفيذ العملية داخل الاقواس حتى لو كانت من مستوى ادنى .

4/ اذا احتوى التعبير على اكثر من قوس واحد سيتم تنفيذ الاقواس الداخلية اولا .

أنواع البيانات Data types

إن البيانات التي نتعامل معها إما أرقام أو حروف او كلمات ، وعندما نقول حرف في لغة سي يمكن ان يكون اي رمز او حرف او رقم موجود على لوحة الكيبورد ، بالنسبة للارقام يمكن ان تكون صحيحة او حقيقية تحتوي على علامة عشرية .
الجدول التالي يوضح انواع البيانات الاكثر شيوعا .

النوع	معناه	مثال	حجم الذاكرة	مدى الأرقام
char	حرف واحد	A 2 *	1	-127 الى+128
int	عدد صحيح	5 3 -5	2	-32768 الى+32768
long	عدد صحيح طويل	5 6	4	الى2014704830648 -2014704830648
float	عدد حقيقي	3.5	4	E-38 الىE+38
double	عدد حقيقي طويل	6.2	8	E-308 الىE+308

المتغير Variable

عبارة عن أسم لمكان يعطى في الذاكرة . الصيغة العامة للأعلان عن متغير هي :

Data type   variable name

Example : int x;   char ch;   long y;   double area;

شروط تسمية المتغير : الأعلان عن المتغيرات التالية خطأ وفق الشروط التالية .

1/ ان لا يبدأ بحرف # . ← int #x;

2/ ان لا يحتوي على فراغ . ← char c h;

3/ ان لا يكون من الكلمات المحجوزة . ← int main;

4/ ان لا يبدأ برقم . ← float 4a;

ملاحظات :

1/ لغة سي تفرق بين الحروف الكبيرة والصغيرة .

2/ يمكن تسمية المتغير بخليط من حروف وارقام على شرط أن تبدأ بالحرف اولا .

كيفية اعطاء المتغير قيمة :

1/ عن طريق التخصيص : x=5;

2/ من ناتج معادلة رياضية : x=y*6+4/20+5;

3/ بإدخال من لوحة المفاتيح .

كيفية الأعلان عن متغير مع اعطاء قيمة أولية :

```
int x=2;
```

كما يمكن الأعلان عن أكثر من متغير في جملة واحدة على شرط ان تكون من نفس النوع

```
int x;
int y;
int z;
```

→

```
int x,y,z;
```

محددات التوصيف

هي عبارة عن حرفين تستخدم في جملة الطباعة (printf) والإدخال (scanf)

لتحديد نوع البيانات المراد طباعتها او إدخالها .

النوع	محدد التوصيف	النوع	محدد التوصيف
char	%c	float	%f
int	%d	double	%f
long	%ld	string	%s

الدالة printf

تستخدم للطباعة على الشاشة . ولها ثلاث صور :

1/ طباعة نص فقط .

2/ طباعة قيمة فقط .

3/ طباعة نص مع قيمة .

تستخدم printf لطباعة النص بشرط أن تكون داخل علامتي التنصيص المزدوجة مع مراعات الاحرف غير المطبوعة (متسلسلات الهروب) .

متسلسلات الهروب :

عبارة عن حرفين تكتب داخل جملة الطباعة printf لا تظهر هذة الحروف ولكن يظهر تأثيرها .

\n	ينقل مؤشر الكتابة الى سطر جديد .
\t	ينقل مؤشر الكتابة الى سبعة خطوات للارقام .
\b	ينقل مؤشر الكتابة حركة للخلف مثل الضغط على مفتاح Back space
\a	اطلاق صوت صافرة . bell

أستخدام printf لطباعة نص مع قيمة :

```
int x=5;

printf("\n x=%d",x);
```

عند التنفيذ سيقوم المترجم بأستبدال الحرفين %d بقيمة x . هذا مخرج البرنامج

```
x=5
```

ملاحظات :

1/ عدد محددات التوصيف يجب ان يساوي عدد المتغيرات .

2/ نوع محددات التوصيف يجب ان يناسب نوع المتغرات .

3/ ترتيب محددات التوصيف يجب ان يناسب ترتيب المتغيرات من اليسار الى اليمين .

امثلة :

```
int x=1,y=6,z=3;

printf("%d\t%d\t%d",x,y,z);

printf("x=%d\ny=%d\nz=%d",x,y,z);
```

```
int x=1;

float y=2.5;

printf("%d\t%f",x,y);

printf("x=%d\ny=%f",x,y);
```

الدالة scanf

تستخدم لإدخال أى قيمة من لوحة المفاتيح .(تدخل أى نوع من البيانات) وتحتاج لمعلومتين :

1/ نوع البيانات المراد إدخالها .

2/ عنوان مكان لنضع فيه البيانات بداخله .

عنوان المتغير :

تستخدم لذلك مؤثر العنوان (&) Address operator هذا المؤثر إذا وضع يسار اسم المتغير يعني عنوانه في الذاكرة . مثال لعنوان : (&x &y &z)

ملاحظات :

1/ عدد محددات التوصيف يجب أن يساوي عدد العناوين .

```
int x,y;
float z;
scanf("%d%d%f",&x,&y,&z);
```

2/ الدالة scanf تسمح بإدخال أى نوع بيانات مخالف بغض النظر عن النوع الذي قمت بتحديده ، لذا لابد من تنبيه المستخدم برسالة بنوع البيانات الذي يجب أن يتم إدخاله .

```
int x;
printf("in put num:");
scanf("%d",&x);
```

```
char a;
printf("in put char");
scanf("%c",&a);
```

Example 1:

أكتب برنامج يقوم بطباعة جملة Welcome C language

```
#include<stdio.h>

#include<conio.h>

void main( )

{

        clrscr( );

        printf("\n welcome C langauge");

        getch( );

}
```

Example 2:

أكتب برنامج يسمح بإدخال عددين صحيحين ومن ثم حساب وطباعة مجموعهما ومتوسطهما .

```
#include<stdio.h>

#include<conio.h>

void main( )

{

        clrscr( );

        int x,y;
```

```
printf("\n in put 2num:");
scanf("%d%d",&x,&y);
printf("\n%d+%d=%d",x,y,x+y);
getch( );
}
```

Example 3:

أكتب برنامج يسمح بإدخال تاريخ الميلاد اليوم / الشهر / السنة ومن ثم طباعة تاريخ الميلاد كاملا وعمرك الحالى .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr;
int x,y,z,age;
printf("please in put date:");
scanf("%d%d%d",&x,&y,&z);
age=2012-z;
printf("\nDate: %d/%d/%d\nyour age is %d",x,y,z,age);
getch( );}
```

Example 4:

أكتب برنامج لحساب مساحة الدائرة .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
float rad,area;
printf("in put radius:");
scanf("%f",&rad);
area=22./7.*rad*rad;
printf("area=%.3f",area);
getch( );
}
```

Example 5:

اكتب برنامج بإستخدام الاعداد الصحيحة يقوم بجمع وطرح وقسمة وضرب وباقي قسمة عددين كلهم في برنامج واحد .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
```

```
clrscr( );
int a,b,sum,sub,prod,div,quot;
a=5;
b=3;
sum=a+b;
sub=a-b;
prod=a*b;
div=a/b;
quot=a%b;
printf("\nThe sum is %d+%d=%d",a,b,sum);
printf("\nThe sub is %d-%d=%d",a,b,sub);
printf("\nThe prod is %d*%d=%d",a,b,prod);
printf("\nThe division is %f /%f=%f",a,b,div);
printf("\nThe quotient is %d,%d=%d",a,b,quot);
getch( );
}
```

Example 6:

أكتب برنامج يسمح بإدخال عددين صحيحين ومن ثم أستبدال الأول بالثاني .

```
#include<stdio.h>

#include<conio.h>

void main( )

{

clrscr( );

int x,y,z;

printf("in put 2num:");
```

```
scanf("%d%d",&x,&y);
z=x;
x=y;
y=z;
printf("\n%d  %d",x,y);
getch( );
}
```

Example 7: ادناه عملية ادخال واخراج تمت بعد تنفيذ برنامج اكتب البرنامج
you type [A] I love [A]

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
char ch;
printf("In put letter:");
scanf("%c",&ch);
printf("\nyou type[%c]I love[%c]",ch,ch);
getch( );}
```

التحكم في سير البرنامج

Control Statement

من النادر ان تجد برنامج مفيد يعبر بالتسلسل من أعلى الى أسفل ، حيث كان البرنامج بسيط ستجد انه ينفذ بعض الجمل ويتخطى الأولى مثل جملة (if)

أو يكرر بعض الأجزاء منه مثل الحلقات التكرارية ، وتوفر لغة سي عدد من جمل التحكم في سير البرنامج .

1/ جملة if

2/ الحلقات loops

3/ جملة exit

4/ جملة break

5/ جملة continuo

6/ تركيب switch

7/ جملة Go to

وغيرها من الجمل ولكل جملة من جمل التحكم هذة مهمة محددة لتنفيذها ، كما يجب إستخدام الجملة الاكثر تناسبا مع البرنامج المراد انشائه لأنه توجد جمل متشابهة في الأداء مثل جملة if و تركيب switch .

جملة If

جملة if لها ثلاثة صور :

1/ if البسيطة .

2/ if ... else

3/ if المتداخلة Nested if

if البسيطة :

الصيغة العامة :

```
 if(condition)
 statement;
or
if(condition)
{
      Statement;
      Statement;
}
```

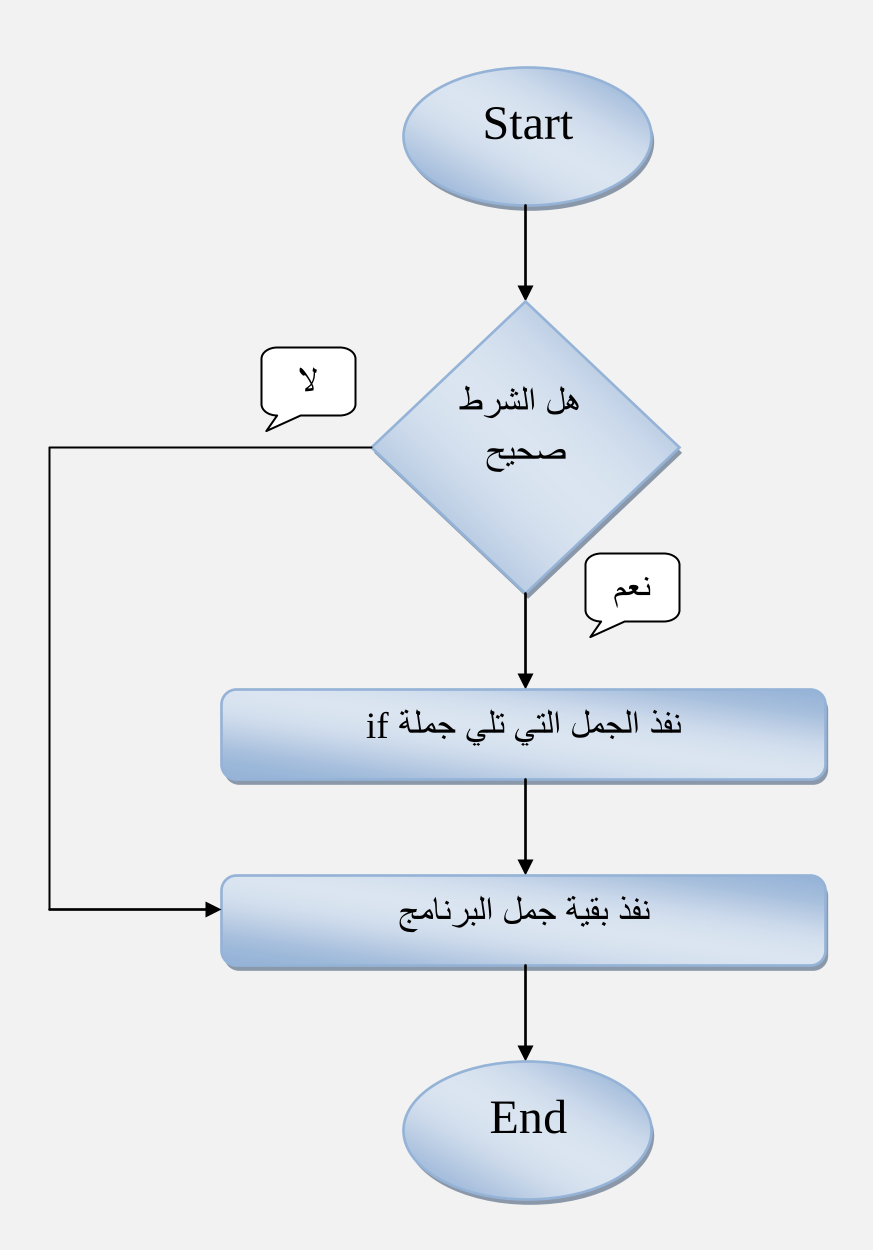
ملاحظات :

1/ جميع جمل التحكم اذا كانت تحتوي على اكثر من جملة بداخلها يجب وضع اقواس .

2/ جمل التحكم في سير البرنامج أغلبها لا تنتهي بفاصلة منقوطة .

كيف تعمل جملة if البسيطة :

إختبر الشرط إذا كان صحيح نفذ الجمل التي تلي جملة if وإذا كان خطأ تجاهل الجمل التي تلي جملة if ونفذ بقية جمل البرنامج .



: if ... else

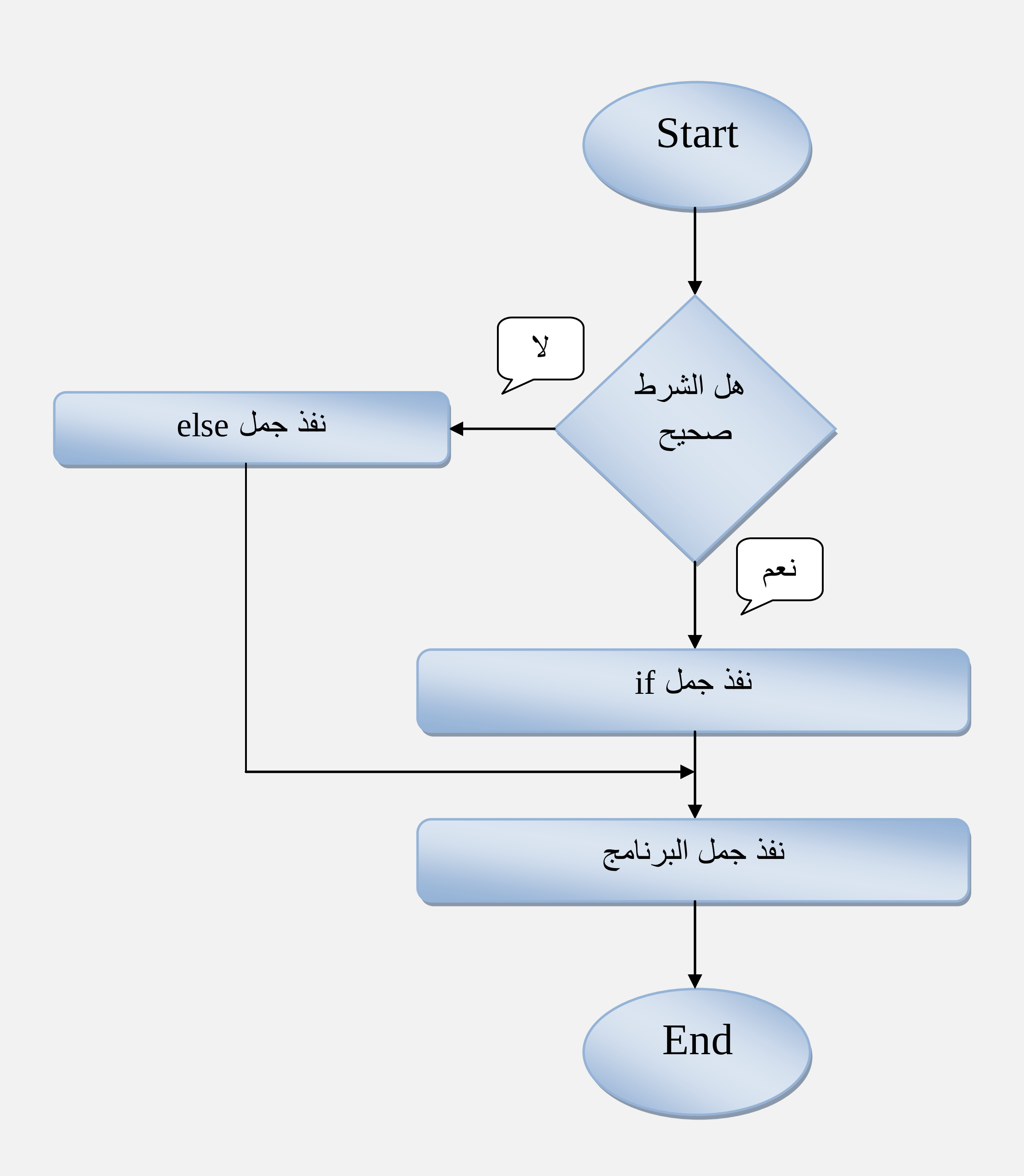
تفيد عند وجود اختيار يمكن أن يأخذ أكثر من احتمال .

الصيغة العامة :

```
if(condition)
        statement;
else
        statement;
or
if(condition)
{
        statement;
        statement;
}
else
{
        statement;
        statement;
}
```

كيف تعمل جملة if ... else :

إذا كان الشرط صحيح نفذ الجمل التي تلي جملة if وتجاهل جمل else وإذا كان الشرط خطأ نفذ جمل else وتجاهل جمل if .



if المتداخلة :

تفيد في اختيار شيء من عدة شروط ، إذا لم تحقق جميع الشروط لن يتم تنفيذ الجمل التي تلي if .

الصيغة العامة :

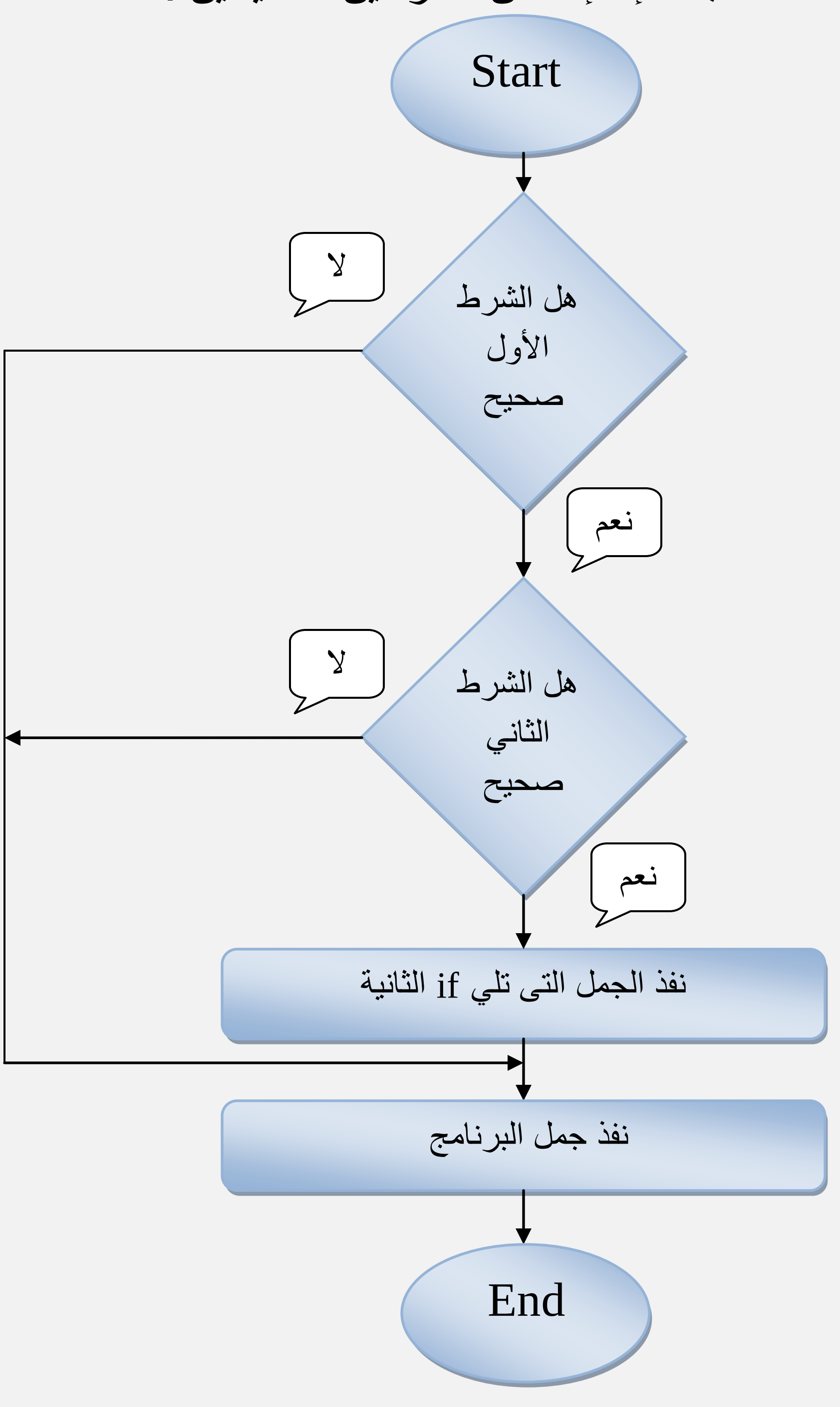
```
if(condition)
if(condition)
        statement;

or

if(condition)
if(condition)
{
        statement;
        statement;
        statement;
}
```

كيف تعمل if المتداخلة :

إذا كان الشرط الأول صحيح والشرط الثاني صحيح نفذ الجمل التي تلي جملة if الثانية ، لن يتم تنفيذ هذة الجمل إلا إذا كان الشرطين صحيحين .



Example 1:

أكتب برنامج يسمح بإدخال عدد صحيح وطباعة إذا كان زوجي أم فردي .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
      clrscr( );
      int x;
      printf("\n in put num:");
      scanf("%d",&x);
      if(x%2==0)
      printf("\n Even %d",x);
      else
      printf("\n odd %d",x);
      getch( );
}
```

Example 2:

أكتب برنامج يسمح بإدخال عددين صحيحين وطباعة أكبرهما أو إذا كانا متساويين.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int x,y;
    printf("\nin put 2num:");
    scanf("%d%d",&x,&y);
    if(x>y)
    printf("\n%d is greater",x);
    else
    if(y>x)
    printf("\n%d is greater",y);
    else
    printf("\nthey one equal %d",x);
    getch( );
```

Example 3:

اكتب برنامج يسمح بإدخال درجة الطالب mark يقوم البرنامج بطباعة التقدير المقابل بعد أختيار الدرجة المدخلة كالأتى :

Excellent A	>=80
Very good B	>=70
Good C	>=60
Pass D	>=50
Fail F	عدا ذلك

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr( );
        int mark;
        printf("\n in put mark:");
        scanf("%d",&mark);
        if(mark>=80)
```

```
        printf("\n Excellent A");
        else
        if(mark>=70)
        printf("\n Very good B");
        else
        if(mark>=60)
        printf("\n Good C");
        else
        if(mark>=50)
        printf("\n Pass D");
        else
        printf("\n Fail F");
        getch( );
}
```

تركيب switch

تفيد في أختيار متغير يمكن أن يأخذ أكثر من قيمة ، وتستخدم أيضا بديل لجملة if ... else .

الصيغة العامة :

```
switch(variable)
{
    case value 1:statement;
    break;
    case value  2:statement;
    break;
    case value 3:statement;
    break;
    default:statement;
}
```

Example 1:

أكتب برنامج يقوم بطباعة one إذا أدخل الرقم 1 وطباعة two إذا أدخل الرقم 2 وطباعة three إذا أدخل الرقم 3 .

```
#include<stdio.h>
#include<conio.h>
```

```
void main( )
{
    clrscr( );
        int x;
printf("\n in put num:");
scanf("%d",&x);
switch(x)
{
case 1:printf("\n one");
break;
case 2:printf("\n two");
break;
case 3:printf("\n three");
break;
default:
printf("\n error");
}
getch( );
}
```

Example 2:

أكتب برنامج يقوم بإجراء العمليات الحسابية على عددين بإستعمال تركيب switch.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
	clrscr( );
	char op;
	float num1,num2;
	printf("\n in put operator:");
	scanf("%c",&op);
	printf("\n in put 2number:");
	scanf("%f%f",&num1,&num2);
switch(op)
{
case '+':
printf("\n%f+%f=%f",num1,num2,num1+num2);
break;
case '-':
```

```
printf("\n%f-%f=%f",num1,num2,num1-num2);
break;
case '*':
printf("\n%f*%f=%f",num1,num2,num1*num2);
break;
case '/':
printf("\n%f/%f=%f",num1,num2,num1/num2);
break;
default:
printf("\nerror...");
}
        getch( );
}
```

الحلقات loops

التكرار :

تنفيذ شيء محدد لعدد محدود من المرات أو حتى يتحقق شرط معين .

التكرار نوعين :

1/ معلوم عدد المرات .

2/ غير معلوم عدد المرات . يجب ان يتحقق شرط لإيقاف التكرار .

جمل التكرار :

1/ for

2/ while

3/ do...while

تكرار for

يفيد تكرار جمل لعدد معلوم من المرات .

الصيغة العامة :

```
for(initial;condition;increase\decrease)
{
    statements;
}
```

يتكون تكرار for من جزأين :

1/ رأس التكرار :

ويتكون من ثلاثة أجزاء تفصل بينهما بفاصلة منقوطة وهي :

أ/ عبارة التمهيد initial : تستخدم لإعطاء متغير التكرار قيمة أولية تنفذ مرة واحدة في بداية التكرار .

ب/ الشرط condition : يمثل شرط استمرار التكرار إذا كان الشرط صحيح يستمر التكرار ويتوقف اذا كان خاطىء وينفذ مع كل دورة تكرار .

ج/ عبارة الزيادة او النقصان : تستخدم لزيادة او انقاص متغير الحلقة بواحد او اى قيمة اخرى ، وتنفذ مرة بعد كل دورة تكرار .

2/ جسم التكرار :

العبارة او الجملة المراد تكرارها . (جملة واحدة بدون قوس عدة جمل بين قوسين)

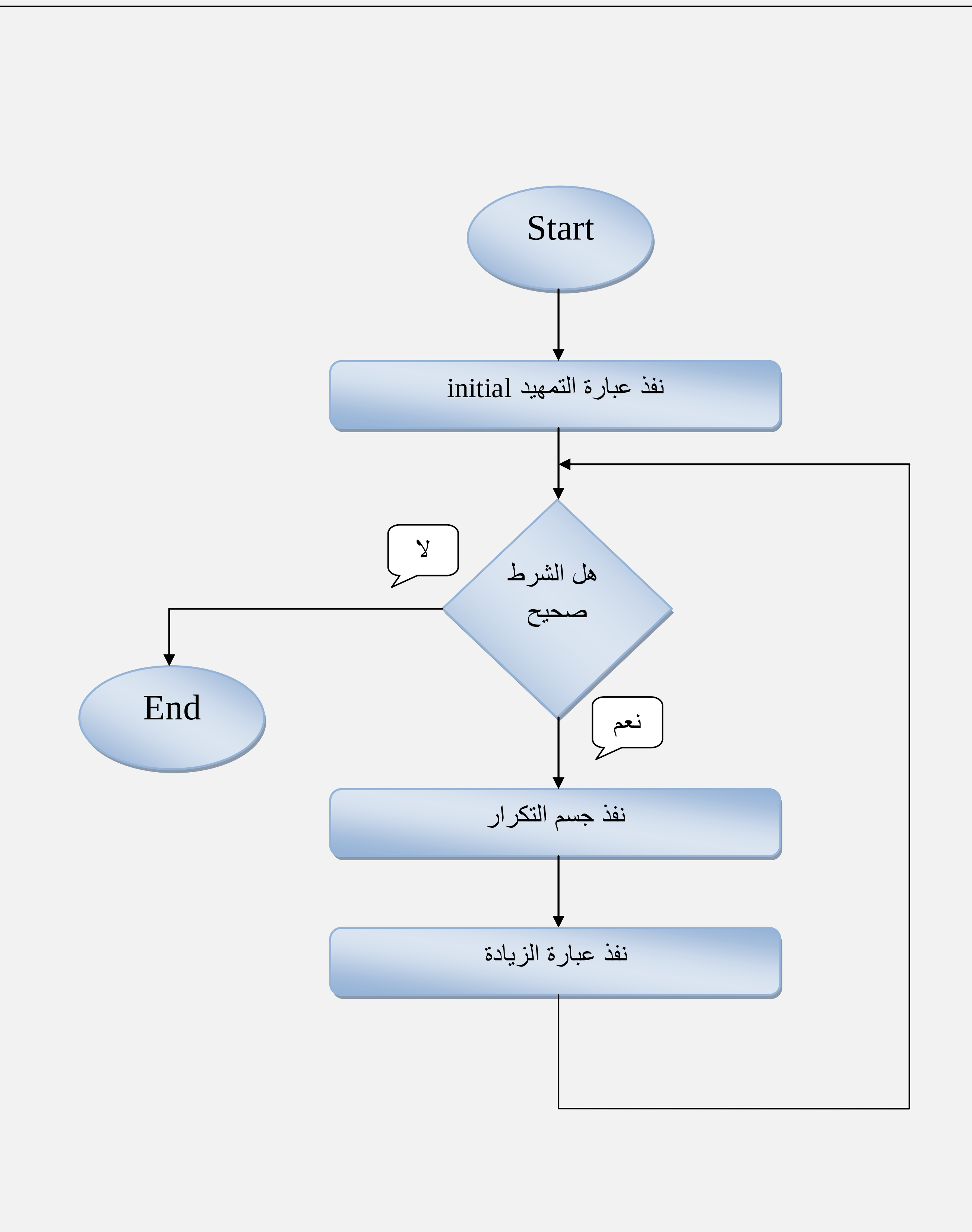
كيف يعمل تكرار for :

1/ نفذ عبارة التمهيد مرة واحدة .

2/ اختبر إذا كان الشرط صحيح نفذ جسم التكرار مرة واحدة .

3/ نفذ عبارة الزيادة او النقصان . يظل التكرار يتنقل ما بين الخطوتين 2 و 3 الى أن يصبح الشرط خاطىء فيتوقف التكرار .

لن يتم تنفيذ التكرار إذا كان الشرط خاطىء منذ البداية .



Start
نفذ عبارة التمهيد initial
هل الشرط صحيح
لا
نعم
End
نفذ جسم التكرار
نفذ عبارة الزيادة

Example 1:

أكتب برنامج يطبع الاعداد من 1 الى 10 .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr( );
        int i;
        for(i=1;i<=10;i++)
        printf("\t%d",i);
        getch( );
}
```

Example 2:

أكتب برنامج يطبع عبارة hello خمسة مرات .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr( );
```

```
    int i;
    for(i=1;i<=5;i++)
    printf("\nhello");
    getch( );
}
```

Example 3:

اكتب برنامج يطبع الاعداد من 1 ½ 3/1 ¼ 5/1 6/1 7/1 .... 10/1 .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    float i;
    for(i=1;i<=10;i++)
    printf("\n%f",1/i);
    getch( );
}
```

تكرار while

يفيد تكرار لعدد غير معلوم من المرات حتى يتحقق شرط معين .

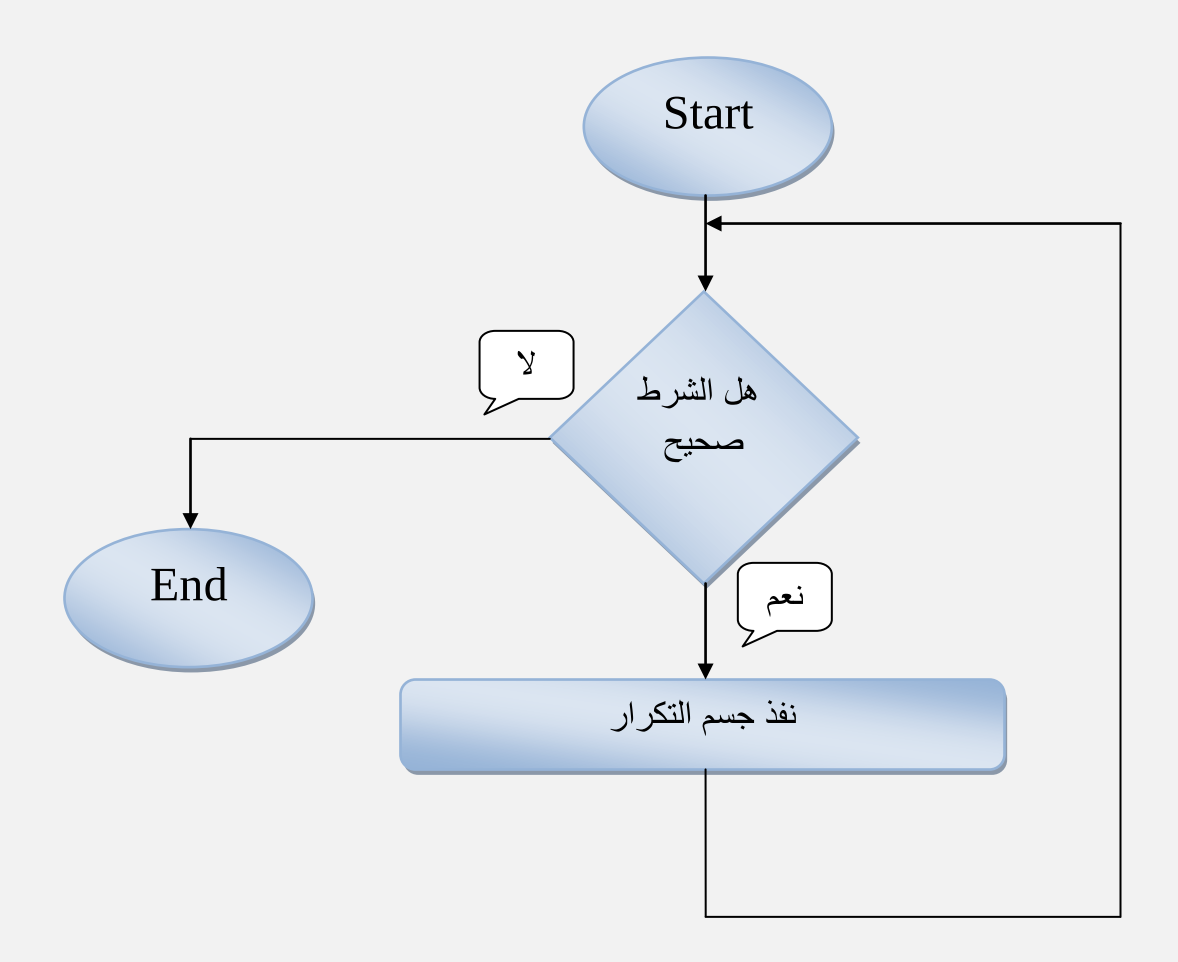
الصيغة العامة :

```
while(condition)
{
        statement 1;
        statement 2;
        statement 3;
}
```

كيف يعمل تكرار while :

اختبر الشرط اذا كان صحيح نفذ جسم التكرار اذا كان خاطىء توقف .

أى قيمة ابتدائية يحتاجها متغير التكرار تكتب قبل الحلقة وقبل بداية التكرار ، وزيادة او انقاص متغير الحلقة تكتب داخل جسم التكرار .



Example 1:

اكتب برنامج يطبع الاعداد من 1 الى 20 بإستخدام جملة while .

```
#include<stdio.h>

#include<conio.h>

void main( )

{

        clrscr( );

        int x=1;
```

```
	while(x<=20)
	{
		printf(" %d",x);
		x++;
	}
	getch( );
}
```

Example 2:

أكتب برنامج يقوم بطباعة جملة Welcome C language عشرة مرات .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
	clrscr( );
	int x=1;
	while(x<=10)
	{
		printf("\nWelcome C Language");
		x++;
```

```
    }
    getch( );
}
```

Example 3:

اكتب برنامج يطبع الاعداد 50 40 30 20 10 .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int x=50;
    while(x>=10)
    {
        printf(" %d",x);
        x=x-10;
    }
    getch( );
}
```

تكرار do...while

يشبه تكرار while يفيد تكرار لعدد غير معلوم حتى يتحقق شرط معين .

الصيغة العامة :

```
do
{
        statement;
}
while(condition);
```

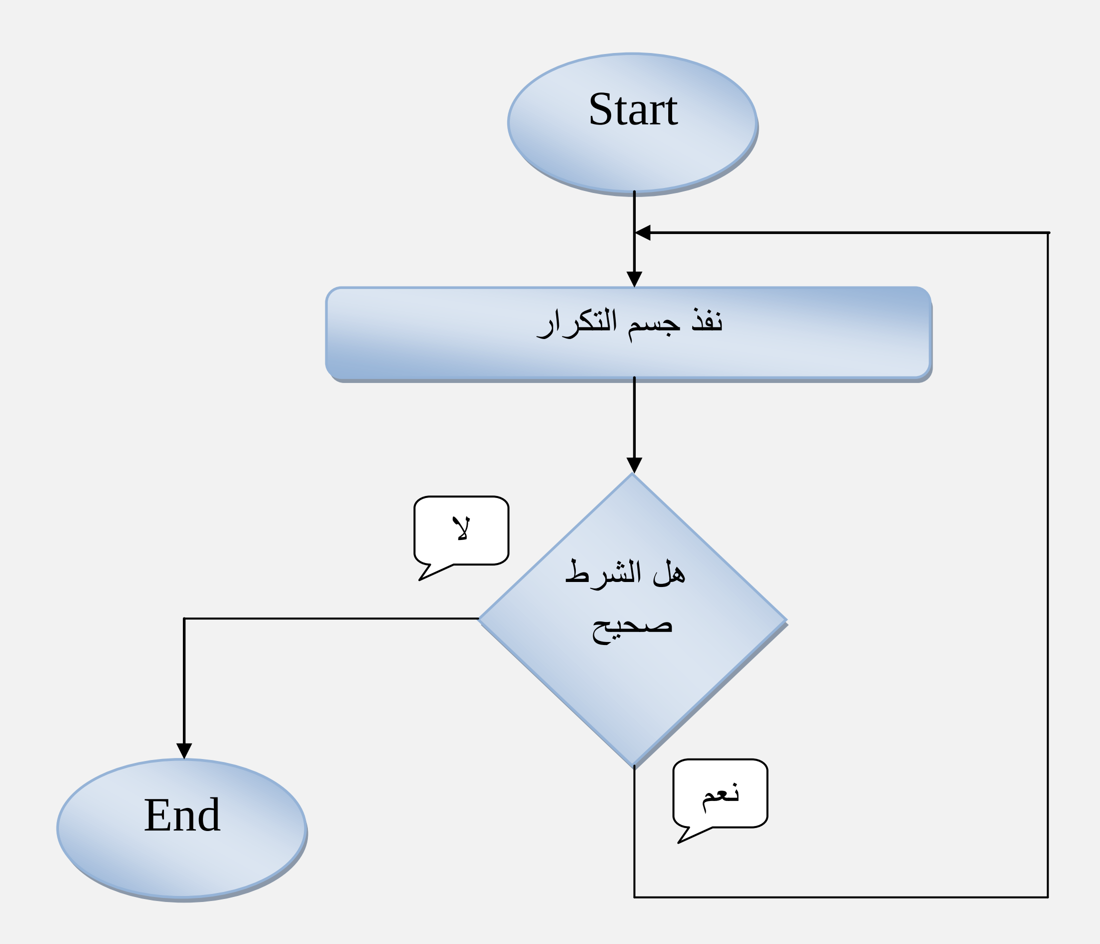
ملاحظة :

يجب وضع الفاصلة المنقوطة بعد شرط while .

كيف يعمل تكرار do...while :

1/ نفذ جسم التكرار مرة واحدة .

2/ أختبر الشرط إذا كان صحيح اذهب الى الخطوة رقم 1 .



ما هو الفرق بين حلقة while و do...while ؟

do...while تنفذ جسم التكرار على الاقل مرة واحدة حتى لو كان الشرط خاطيء منذ البداية لانه يختبر الشرط بعد تنفيذ جسم التكرار .

Example 1:

اكتب برنامج يطبع الاعداد 5 4 3 2 1 بإستخدام do...while .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int a=5;
    do
    {
        printf("\n%d",a);
        a--;
    }
    while(a>=1);
    getch( );
}
```

Example 2:

اكتب برنامج يقوم بإدخال عدد صحيح ومن ثم طباعة العدد ومربعه ومكعبه ، وإذا ادخل عدد اكبر من 100 يتم الخروج من البرنامج .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
	clrscr( );
	int x;
	do
	{
		printf("\nin put num :");
		scanf("%d",&x);
		printf("\n\n%d\t%d\t%d",x,x*x,x*x*x);
	}
	while(x<=100);
	getch( );
}
```

الحلقات المتداخله Nested loops

أن تكون هنالك حلقة داخل حلقة أخرى .

كيف تعمل الحلقات المتداخلة :

سيتم تنفيذ الحلقة الداخليه أولا ثم العودة الى الحلقة الخارجية لعبارة الزيادة او النقصان ثم اختبار الشرط ثم العودة الى الحلقة الداخليه الى ان تنتهي الحلقة الخارجية .

Example 1:

اكتب برنامج لطباعة جداول الضرب من جدول 1 الى جدول 12 بإستعمال حلقة

for .

```
#include<stdio.h>

#include<conio.h>

void main( )

{

clrscr( );

int i,j;

for(i=1;i<=12;i++){

printf("\n\ntable[%d]",i);

for(j=1;j<=12;j++)

printf("\n%d*%d=%d",i,j,i*j);
```

```
getch( );}
}
```

إستخدام الحلقات المتداخلة لطباعة الاشكال الهندسية :

Example 1:

اكتب برنامج لطباعة الشكل :

```
****
****
****
****
```

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int i,j;
    for(i=1;i<=4;i++)
    {
        for(j=1;j<=4;j++)
        printf("*");
        printf("\n");
    }
    getch( );}
```

Example 2:

اكتب برنامج لطباعة الشكل :

```
*
**
***
****
```

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int i,j;
    for(i=1;i<=4;i++)
    {
        for(j=1;j<=i;j++)
        printf("*");
        printf("\n");
    }
    getch( );
}
```

Example 3:

اكتب برنامج لطباعة الشكل :

```
****
***
**
*
#include<stdio.h>

#include<conio.h>

void main( )

{

    clrscr( );

    int i,j;

    for(i=1;i<=4;i++)

    {

        for(j=i;j<=4;j++)

        printf("*");

        printf("\n");

    }

    getch( );

}
```

Example 4:

اكتب برنامج لطباعة الشكل :

```
   *
  **
 ***
****
```

```
#include<stdio.h>

#include<conio.h>

void main( )

{

    clrscr( );

    int i,j,k;

    for(i=1;i<=4;i++)

    {

        for(k=1;k<=4-i;k++)

        printf(" ");

        for(j=1;j<=i;j++)

        printf("*");

        printf("\n");

    }

    getch( );

}
```

Example 4:

اكتب برنامج لطباعة الشكل :

```
****
 ***
  **
   *
```

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int i,j,k;
    for(i=1;i<=4;i++)
    {
        for(k=1;k<=i-1;k++)
        printf(" ");
        for(j=1;j<=5-i;j++)
        printf("*");
        printf("\n");
    }
    getch( );
}
```

المصفوفات Arrays

المصفوفة هي مجموعة من العناصر تنتمي الى نوع واحد من انواع البيانات ، ويخصص لها اسم واحد وتنقسم المصفوفات الى مصفوفات ذات بعد واحد ومصفوفات ذات بعدين .

1/ المصفوفة ذات البعد الواحد :

تسمى مصفوفة ذات بعد واحد لأنها تتكون من صف واحد أو عمود واحد . ويتم الإشارة إلى كل عنصر في المصفوفة بترتيبه داخل المصفوفة على أن يبدأ العد بالرقم صفر .

الأعلان عن المصفوفة ذات البعد الواحد :

```
Data type    Array name[size];
```

```
Ex1: int A[5];
```

```
Ex2: int A[4]={5,8,1,4};
```

في المثال الأول تم الاعلان عن مصفوفة ذات بعد واحد من نوع صحيح وعدد عناصرها خمسة . وفي المثال الثاني تم الإعلان عن مصفوفة ذات بعد واحد من نوع صحيح عدد عناصرها أربعة وتم وضع قيمة ابتدائية لكل عنصر حيث يرمز لكل عنصر كالتالي .

العنصر الأول : A[0]=5

العنصر الثاني : A[1]=8

العنصر الثالث : A[2]=1

العنصر الرابع : A[3]=4

Example 1:

اكتب برنامج يقوم بطباعة عناصر المصفوفة التي تم الاعلان عن عناصرها وأعطيت قيم ابتدائية .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int i,A[4]={1,2,3,4};
for(i=0;i<=3;i++)
printf("\n%d",A[i]);
getch( );
}
```

Example 2:

اكتب برنامج يقوم بطباعة العنصر الثاني من المصفوفة التي تم استقبال عناصرها من المستخدم وهي خمسة عناصر .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
```

```
clrscr( );
int i,A[5];
for(i=0;i<=4;i++)
{
printf("\nA[%d]=",i);
scanf("%d",&A[i]);
}
printf("\nA[1]=%d",A[1]);
getch( );
}
```

Example 3:

اكتب برنامج يقوم بإدخال قيم لعناصر المصفوفة ذات البعد الواحد وبعدها عشرة ، ومن ثم طباعة مجموع العناصر ومتوسطها .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int i,sum=0,A[10];
```

```
float average;
for(i=0;i<10;i++)
{
printf("\nA[%d]=",i);
scanf("%d",&A[i]);
sum=sum+A[i];
}
average=sum/10.;
printf("\nsum=%d\naverage=%f",sum,average);
getch( );
}
```

2/ المصفوفة ذات البعدين :

هي المصفوفة التي ترتب عناصرها في شكل صفوف وأعمدة ويتم الاعلان عنها بالشكل التالي :

```
int A[4][5];
float [5][10];
```

يتم الإشارة الى العنصر برقم الصف ورقم العمود ، ويجب الانتباه الى أنه عندما تستخدم مصفوفة لا بد من استعمال الدوارة for كما في المصفوفة ذات البعد الواحد

أما في حالة المصفوفة ذات البعدين فلا بد من استعمال الحلقات المتداخلة .

Example 1:

أكتب برنامج يقوم باستقبال مجموعة قيم ويخزنها في مصفوفة ذات بعدين ثم يقوم بطباعة هذة القيم ومن ثم طباعة مجموعها .

```
#include<stdio.h>

#include<conio.h>

void main( )

{

clrscr( );

int x,y,sum=0;

int A[3][4];

for(x=0;x<3;x++)

{

printf("\n");

for(y=0;y<4;y++)

{

printf("A[%d][%d]=",x,y);

scanf("%d",&A[x][y]);

sum=sum+A[x][y];

}
```

```
}
clrscr( );
for(x=0;x<3;x++)
{
printf("\n");
for(y=0;y<4;y++)
printf("\nA[%d][%d]=%d",x,y,A[x][y]);
}
printf("\nsum=%d",sum);
getch( );
}
```

Example 2:

أكتب برنامج يقوم بطباعة عناصر المصفوفة في شكل مصفوفة ذات بعدين التي تم الأعلان عنها واعطيت قيم ابتدائية وبعدها 4×3 .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr( );
```

```
int x,y,A[3][4]={{5,3,7,2},{5,3,1,9},{6,3,7,9}};
for(x=0;x<3;x++)
{
    printf("\n");
    for(y=0;y<4;y++)
    printf("\t%d",A[x][y]);
}
getch( );
}
```

الدوال Functions

الدوال التي استخدمناها سابقا مثل printf و scanf تعتبر دوال مكتبية أى تتبع للغة

سي وهي دوال عامة يستطيع أى مبرمج استخدامها مما يظهر مزايا ومرونة هذة اللغة . والدالة عبارة عن برنامج صغير أو مجموعة تعليمات تؤدي غرض معين يخصص لهذا البرنامج اسم ويتم استدعائه داخل الدالة الرئيسية main .

مزايا استخدام الدوال :

1/ عدم تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحدة ثم يتم استدعائها أكثر من مرة عند الحاجة اليها .

2/ باستخدام الدوال يصبح البرنامج أكثر وضوحا .

الصيغة العامة :

```
Data type   function name(data type  variable,data type  variable)

{

return;

}
```

Ex1:

```
int add(int a,int b)

{

return(a+b);

}
```

أنواع الدوال :

1/ دوال تعيد قيمة صحيحة int function .

2/ دوال تعيد قيمة حقيقية float function .

3/ دوال تعيد عبارة حرفية string function .

4/ دوال تعيد حرف واحد char function .

5/ دوال لا تعيد أي قيمة void function .

6/ دوال تعيد قيمة من نوع structure .

Example 1:

اكتب برنامج بإستخدام الدوال لجمع عددين .

```
#include<stdio.h>
#include<conio.h>
int add(int a,int b)
{
return(a+b);
}
void main( )
{
clrscr( );
int x,y;
```

```
printf("in put 2num:");
scanf("%d%d",&x,&y);
printf("the sum=%d",add(x,y));
getch( );
}
```

Example 2:

اكتب برنامج بإستخدام الدوال يقوم بإجراء العمليات الحسابية على عددين صحيحين

```
#include<stdio.h>
#include<conio.h>
int sum(int a,int b)
{
        return(a+b);
}
int sub(int a,int b)
{
        return(a-b);
}
int mul(int a,int b)
```

```
{
        return(a*b);
}
int div(int a,int b)
{
        return(a/b);
}
void main( )
{
        clrscr( );
int x,y;
printf("in put 2num:");
scanf("%d%d",&x,&y);
printf(" summation=%d\n subtraction=%d",sum(x,y),sub(x,y));
printf("\n multiplication=%d\n division=%d",mul(x,y),div(x,y));
getch( );
}
```

Example 3:

اكتب برنامج بإستخدام الدوال يقوم بإدخال اربعة اعداد صحيحة وطباعة اكبرهما .

```
#include<stdio.h>
#include<conio.h>
int osm(int q)
{
int a,b,c,d;
printf("\nin put 4num:");
scanf("%d%d%d%d",&a,&b,&c,&d);
if(a>b&&a>c&&a>d)
printf("\n%d is greater",a);
else
if(b>a&&b>c&&b>d)
printf("\n%d is greater",b);
else
if(c>a&&c>b&&c>d)
printf("\n%d is greater",c);
else
if(d>a&&d>b&&d>c)
```

```
printf("\n%d is greater",d);
        return(q);
}
void main( )
{
        clrscr( );
int q;
printf("\n%d",osm(q));
getch( );
}
```

المختصر Macro

هو مجموعة من التعليمات التي تؤدي وظيفة محددة وهو يشبه الدوال في وظائفها ، وايضا يتم إنشائه مرة واحدة ويتم استدعائه داخل الدالة الرئيسية كلما احتجنا اليه .

كيفية انشاء الماكرو :

يتم انشائه باستخدام التوجيه #define .

الصيغة العامة :

```
#define macro
Ex1: #define sum(x,y)x+y
Ex2: #define sub(a,b)a-b
```

Example 1:

أكتب برنامج باستخدام الماكرو يقوم بجمع عددين .

```
#define sum(a,b)a+b
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int v1,v2;
```

```
    printf("\nin put 2num:");
    scanf("%d%d",&v1,&v2);
    printf("\nthe summation = %d",sum(v1,v2));
    getch();
}
```

Example 2:

اكتب برنامج باستخدام الماكرو يقوم بإدخال ثلاثة اعداد وحساب (x+y)/z .

```
#define eq(a,b,c)(a+b)/c
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int x,y,z;
    printf("\nin put 3num:");
    scanf("%d%d%d",&x,&y,&z);
    printf("\n(%d+%d)/%d=%d",x,y,z,eq(x,y,z));
    getch( );
}
```

Structures السجلات

تطبيقات قواعد البيانات هي من أهم التطبيقات فمثلاً قاعدة بيانات موظفين تمثل بيانات الموظفين في شكل سجلات كل سجل يتكون من مجموعة من الحقول .

كيفية انشاء سجل :

Example 1:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
    struct data
    {
        char stat;
        int num;
    };
    struct data stud;
    stud.num=5;
    stud.stat='t';
    printf("\n stud.num=%d",stud.num);
```

الصيغة العامة لإنشاء سجل حيث stat و num تمثل محتويات او بيانات السجل struct data .

اعلان عن سجل اسمه stud

بيانات السجل الذي اسمه stud . يجب وضع نقطة بين اسم السجل و بياناته .

```
        printf("\t stud.stat=%c",stud.stat);
        getch( );
}
```

Example 2:

اكتب برنامج بإستعمال السجلات يحتوي على شخصين ومن ثم طباعة حرف ورقم الشخصين .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr( );
        struct data
        {
                char stat;
                int num;
        };
        struct data person1,person2;
        person1.num=5;
        person1.stat='a';
```

```
        person2.num=10;
        person2.stat='b';
        printf("\n person1.num=%d",person1.num);
        printf("\t person1.stat=%c",person1.stat);
        printf("\n person2.num=%d",person2.num);
        printf("\t person2.stat=%c",person2.stat);
        getch( );
}
```

Example 3:

قم بإنشاء سجل لشخص يتم ادخال بياناته من المستخدم ومن ثم طباعة بياناته .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
struct data
{
char name[10];
int num,mark;
```

```
};
struct data person;
printf("\n in put name of person :");
scanf("%s",&person.name);
printf("\n in put number of person :");
scanf("%d",&person.num);
printf("\n in put mark of person :");
scanf("%d",&person.mark);
printf("\n %d\\ %s =
%d%",person.num,person.name,person.mark);
getch( );
}
```

Example 4:

اكتب برنامج يقوم بوضع محتويات سجل مدخلة بواسطة المستخدم في سجل اخر .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    clrscr( );
```

```
struct data
{
int no;
char name[10];
};
struct data stud1,stud2;
printf("\nstud1.no:");
scanf("%d",&stud1.no);
printf("\nstud1.name:");
scanf("%s",&stud1.name);
stud2=stud1;
printf("\n stud1.no:%d\n\tstud1.name:%s",stud1.no,stud1.name);
printf("\n stud2.no:%d\n\tstud2.name:%s",stud2.no,stud2.name);
getch( );
}
```

Nested Structures السجلات المتداخلة

هي أن يكون هناك سجل بداخله سجل أو مجموعة سجلات ، بمعنى أن تكون العناصر أو بعضها سجلات وهذا ما يسمى بالسجلات المتداخلة .

Example 1:

اكتب برنامج باستخدام السجلات المتداخلة لمجموعة تحتوي على شخصين وكود للمجموعة ولكل شخص رقم و اسم ، نقوم نحن بإدخال بيانات الشخص الأول ومن ثم يقوم البرنامج بمساوات بيانات الشخص الثاني بالشخص الأول ، ثم يقوم البرنامج بطباعة كل البيانات المدخلة .

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
struct person
{
int no;
char name[10];
};
struct group
{
```

```
struct person p1;

struct person p2;

int code;

};

struct group g1;

printf("\ng1.p1.no=");

scanf("%d",&g1.p1.no);

printf("\ng1.p1.name=");

scanf("%s",&g1.p1.name);

printf("\ng1.code=");

scanf("%d",&g1.code);

g1.p2=g1.p1;

printf("\n\t\tthe data of the group:");

printf("\n\tg1.p1.no=%d\tg1.p1.name=%s",g1.p1.no,g1.p1.name
);

printf("\n\tg1.code=%d",g1.code);

printf("\n\tg1.p2.no=%d\tg1.p2.name=%s",g1.p2.no,g1.p2.name
);

getch( );

}
```

المؤشرات Pointers

المؤشر هو نوع من انواع البيانات ويعتبر بأنه متغير يحتفظ بعنوان مكان في الذاكرة .

من المعلوم أن كل مكان في الذاكرة له عنوان والجهاز يتعامل مع هذا المكان بالعنوان المحدد له ونحن بطريقة غير مباشرة نتعامل مع هذا العنوان ، فمثلاً هذا الإعلان ;int a=5 معناه احجز مكان في الذاكرة حجمه 2 بايت (حجم int) واجعل

اسمه a وضع فيه القيمة 5 ، وبالتالي كلما تعاملنا مع المتغير a فنحن نتعامل مع

القيمة المخزنة فيه وليس العنوان المخصص لهذة القيمة ، هذا عن الإعلان العادي فماذا عن الإعلان عن المؤشر .

كيفية الإعلان عن المؤشر :

يتم الاعلان عن المؤشر بنفس الطريقة التي نعلن بها عن البيانات العادية ولكن الفرق هو ان اسم المؤشر يسبق بالعلامة * ليدل على أنه مؤشر . فمثلاً للإعلان عن مؤشر من نوع صحيح نكتب الصورة التالية :

```
int *p;
```

هذا الإعلان يعني أن المتغير p أصبح مؤشر إلى مساحة في الذاكرة مقدارها 2 بايت

مع الاحتفاظ بعنوان هذا المكان في المتغير p .

وكلما أردنا أن نتعامل مع هذة القيمة تعاملنا عن طريق العنوان أي بدلاً من أن نتعامل مع القيمة ونترك الجهاز يتعامل مع العنوان بهذا الأسلوب نستطيع أن نتعامل مباشرة مع عنوان المكان مما يعطينا القدرة على عمليات كثيرة منها التعامل مع مخارج الجهاز مثل مخرج الة الطباعة حيث أن لمخرج الطابعة عنوان فنستطيع أخذ هذا العنوان وتخزينه في متغير ثم التعامل مع هذا المتغير كما نشاء وكذلك الكتابة في ذاكرة العرض مباشرة .

مزايا استخدام المؤشرات :

يحقق استعمال المؤشرات فوائد كثيرة منها :

1/ إعادة أكثر من قيمة من الدوال .

2/ التعامل مع المصفوفات وتمريرها الى الدوال بشكل افضل .

3/ إنشاء أنواع أكثر قوة من البيانات .

4/ التعامل مع الجهاز ومكوناته وعناوين مداخل ومخارج الجهاز .

من الفوائد المشهورة للمؤشرات استخدامها في إعادة أكثر من قيمة من الدالة .
شرحنا الدوال والتعامل معها وكيفية إعادة قيمة من الدالة لاحظنا سابقاً في الأمثلة التي استخدمناها أننا استخدمنا كلمة return مرة واحدة مع كل دالة وهذا معناه عدم

إمكانية إعادة أكثر من قيمة من الدالة . لو فرضنا أن لدينا مجموعة عمليات وأردنا إنشاء دالة لهذة العمليات وأنشأنا الدالة وتم حساب نتائج العمليات ووضعت هذة النتائج في متغيرات وأردنا إعادة هذة القيم إلى الدالة الرئيسية هنا تظهر المشكلة ، وهي أننا لا نستطيع استعمال أكثر من كلمة وكلمة return لا تعيد إلا قيمة واحدة أما

في حالة استخدام المؤشرات فيمكننا إعادة أكثر من قيمة .

Example 1:

اكتب برنامج باستعمال المؤشرات والدوال يقوم بإعادة قيمتين من الدالة ومن ثم يقوم بطباعة العددين الصحيحين .

```
#include<stdio.h>

#include<conio.h>

void get2(int *xx,int *yy)

{
```

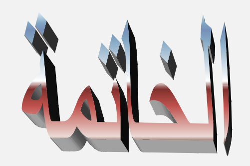
```
        *xx=*xx+5;
        *yy=*yy+10;
}
void main( )
{
        clrscr( );
        int x=5,y=10,*p1,*p2;
        p1=&x,p2=&y;
        get2(p1,p2);
        printf("\nfirst no is %d\tsecond no is %d",x,y);
        getch( );
}
```

Example 2:

اكتب برنامج بإستعمال المؤشرات يقوم بإعادة ثلاثة قيم من دالة صحيحة .

```
#include<stdio.h>
#include<conio.h>
int all(int *a,int *b,int *c)
{
        *a=*a+*b;
```

```
        *b=*a-*b;
        *c=*a**b;
}
void main( )
{
        clrscr( );
        int x=3,y=2,z=1,*x1,*y1,*z1;
        x1=&x,y1=&y,z1=&z;
        all(x1,y1,z1);
        printf("\n%d\t%d\t%d",x,y,z);
        getch( );
}
```



هذا الكتاب المتواضع الذي تناولنا فيه اساسيات لغة السي بإذن الله سبحانه وتعالى سيكون عوناً لكم في دراسة هذة اللغة المهمة جداً في عالم البرمجة وأن هذة اللغة تعتبر من أكثر اللغات مرونتاً ، فإذا أردت أن تجيد هذة اللغة فما عليك إلا أن تطلع على الكثير من البرامج وكتابتها . كما أنه توجد كتب الكترونية أخرى تشتمل على برامج فقط يمكنك تحميلها من الموقع التالي .

http://www.kutub.info

وفي النهاية فما علينا إلى أن نذكركم أن مصدر معلومات هذا الكتاب هي من دراستي الجامعية والأنترنت ، وأن هذا الكتاب توجد منه تحديثات يمكنك التحديث من الموقع السابق أو التالي عند ظهور النسخة القادمة بإذن الله ، وأن هذة التحديثات لا تختلف كثيراً عن بعضها إلا أنه سيتم اضافة وتعديل بعض الأشياء في هذا الكتاب .

For Update :

http://4shared.com/q/CCAD/1/C programing language 5.0

لمزيد من الاستفسارات نرجو المراسلة على البريد الألكتروني :

- osman.abdelmonim@gmail.com

تم بحمد الله